The following is a brief explanation of an example embodiment of the invention, and one of the many ways an example embodiment can be read on a claim, in order to assist the Examiner in understanding the invention.

Briefly, a present invention embodiment is directed toward a virtual supercomputer that is implemented by a computer system. In other words, even though the computer system includes physical hardware slower than that of a supercomputer, the computer system may still provide a solution to certain complex problems in approximately the same amount of time as a supercomputer (e.g., thereby implementing a virtual supercomputer). For example, if a supercomputer takes an hour to arrive at a solution to a problem, the computer system implementing the virtual supercomputer according to an embodiment of the present invention may take approximately the same or lesser amounts of time to arrive at that solution.

The computer system includes software that implements a reconfigurable (virtual) hardware processor (e.g., NVSI Virtual Machine as viewed in Fig. 1) and an associated (virtual) operating system (e.g., NVSI-OS as viewed in Fig. 1). Thus, the computer system emulates a physical processor or hardware architecture (e.g., including its own instruction set (e.g., See Substitute Specification Page 45, Table I)) that can actually be realized in hardware. However, since the physical processor is implemented by a software emulation, the hardware architecture may be configurable to be tailored to the specific problem or application. The reconfiguration of the hardware processor is one of the key aspects of the embodiment.

The virtual supercomputer includes a set of operations and procedures that allow the processor architecture being emulated to be easily tailored and adapted to specific problems or classes of problems. The problems solved by the virtual supercomputer include a solution space

represented by a node structure, where at least one node includes data for the problem and the nodes are traversed and processed to determine a solution to the problem. The emulated hardware architecture includes a design specifically tailored for processing the nodes within the solution space (i.e., a Non-Von Neumann architecture; See instruction set with node operations, Substitute Specification Page 45, Table I)), as opposed to register based processing (i.e., Von Neumann architectures) of conventional architecture schemes.

An example application pertains to computing portfolio risk (e.g., See Substitute Specification Page 55 (First Paragraph)). Initially, the problem space is defined by a user application, and populated with numerous nodes. The nodes represent a range of financial instruments in a portfolio and include pricing information. Once the solution space is populated, the portfolio can be marked to a future state by navigating to that state (or through the portfolio nodes) and applying the pricing information (or pricing vectors) to each instrument. In this fashion, various scenarios can be searched for extreme events (e.g., See Substitute Specification Page 58 (Second Paragraph)).

The domain or user application (e.g., Domain Application as viewed in Fig. 1) defines the problem and processing for the virtual operating system (e.g., NVSI-OS as viewed in Fig. 1) that manages the node structure and controls populating the nodes with data and processing the nodes by the virtual machine (e.g., NVSI Virtual Machine as viewed in Figs. 1 and 2) to determine a solution. The virtual operating system includes an instantiation engine (e.g., IE as viewed in Fig. 1) that provides instructions (e.g., See Substitute Specification Page 45, Table I) for the instantiation unit (e.g., IU as viewed in Fig. 2) of the virtual machine to create and delete the nodes in the solution space. A population engine (e.g., PE as viewed in Fig. 1) of the virtual operating system provides instructions (e.g., See Substitute Specification Page 45, Table I) for

the population unit (e.g., PU as viewed in Fig. 2) of the virtual machine to store data into the nodes in the solution space. A navigation engine (e.g., NE as viewed in Fig. 1) of the virtual operating system provides instructions (e.g., See Substitute Specification Page 45, Table I) for the navigation unit (e.g., NU as viewed in Fig. 2) of the virtual machine to traverse the solution space and read and process selected nodes. An evolution engine (e.g., EE as viewed in Fig. 1) of the virtual operating system provides instructions (e.g., See Substitute Specification Page 45, Table I) for updating the contents of the instantiation and population units (e.g., IU and PU as viewed in Fig. 2) of the virtual machine. A configuration engine (e.g., CE as viewed in Fig. 1) of the virtual operating system provides instructions (e.g., See Substitute Specification Page 45, Table I) for a solution-space configuration unit (e.g., SCU as viewed in Fig. 2) of the virtual machine to create a solution space topology for the specific problem (e.g., See Substitute Specification Page 8).

The virtual operating system interacts with the virtual machine via a virtual assembler (e.g., NVCL Assembler as viewed in Fig. 1) analogous to a conventional assembler or compiler that converts instructions into commands the virtual machine understands. The virtual machine interacts with platform drivers (e.g., Platform Drivers as viewed in Fig. 1) that interact with the underlying operating system of the host computer system performing the emulation (e.g., via a Platform Assembler as viewed in Fig. 1) (e.g., See Substitute Specification Pages 8 - 9).

Accordingly, operation of the system is initiated by a user application defining the problem for the virtual operating system. The virtual operating system subsequently creates a node structure with each node containing information for the problem, while the virtual machine navigates the node structure to process the node information and arrive at a solution to the problem.

The following is one of the many ways an example embodiment can be read on a claim.

4(New). A system to determine solutions for problems including a solution space represented by one or more nodes, said system comprising:

a computer system to dynamically configure and emulate a hardware architecture of a processing system to determine a solution for a problem including a solution space represented by one or more nodes with at least one node including data for said problem,

-> *This may represent the underlying physical computer system emulating the processor and operating system to provide a solution to a problem (e.g., See Figs. 1 and 2; Substitute Specification Pages 4 (Last Paragraph), 6 (Second Paragraph), 7 (Last Paragraph), 19 (Last Paragraph), 21("Solution Space" and "Node"), and 57 (Second Paragraph)).*

wherein said computer system includes:

an operating system to control operation of said computer system;

-> *This may represent the operating system of the underlying physical computer system (e.g., See Fig. 1, Platform CPU/OS; Substitute Specification Page 9 (First Paragraph)).*

a virtual machine unit to emulate said hardware architecture and manage said nodes within said solution space, wherein said hardware architecture is based on processing said nodes;

-> This may represent the underlying computer system emulating or implementing the virtual machine (e.g., See Fig. 1, NVSI-Virtual Machine, and Fig. 2; Substitute Specification Pages 6 (Second Paragraph), 9 (First Paragraph), and 23 (First Paragraph)).

a virtual operating system to configure said hardware architecture and to control operation of said virtual machine unit to emulate said hardware architecture in accordance with a user software application defining said problem and corresponding processing to determine said solution,

-> This may represent the computer system emulating or implementing the operating system of the virtual machine (e.g., See Fig. 1, NVSI-OS; Substitute Specification Pages 8 (Last Paragraph), 23 (First Paragraph), and 30 (First Paragraph)).

said virtual operating system including:

an instantiation engine to create and delete said nodes;

-> This may represent the computer system implementing the instantiation engine of the virtual operating system (e.g., See Fig. 1, IE of the NVSI-OS; and Substitute Specification Pages 8 (First Paragraph), 13 (Second Paragraph), 24 ("Instantiation Unit"), and 77 ("Instantiation Manager")).

a configuration engine to configure said nodes of said solution space in a topology suitable for determining said solution for said problem;

-> This may represent the computer system implementing the configuration engine of the virtual operating system (e.g., See Fig. 1, CE of the NVSI-OS; and Substitute Specification Page 8 (First and Second Paragraphs)).

a population engine to store and evaluate said data for said problem within said nodes;

-> This may represent the computer system implementing the population engine of the virtual operating system (e.g., See Fig. 1, PE of the NVSI-OS; and Substitute Specification Pages 8 (First Paragraph), 13 (Third Paragraph), 24 ("Population Unit"), and 77 ("Population Manager")).

a navigation engine to traverse said topology and process selected ones of said nodes in accordance with said user software application to determine said solution; and

-> This may represent the computer system implementing the navigation engine of the virtual operating system (e.g., See Fig. 1, NE of the NVSI-OS; and Substitute Specification Pages 8 (First Paragraph), 14 (First Paragraph), 25 ("Navigation Unit"), and 77 ("Navigation Manager")).

an evolution engine to update said nodes and said topology in accordance with said user software application.

-> This may represent the computer system implementing the evolution engine of the virtual operating system (e.g., See Fig. 1, EE of the NVSI-OS; and Substitute Specification Page 14 (Second Paragraph)).